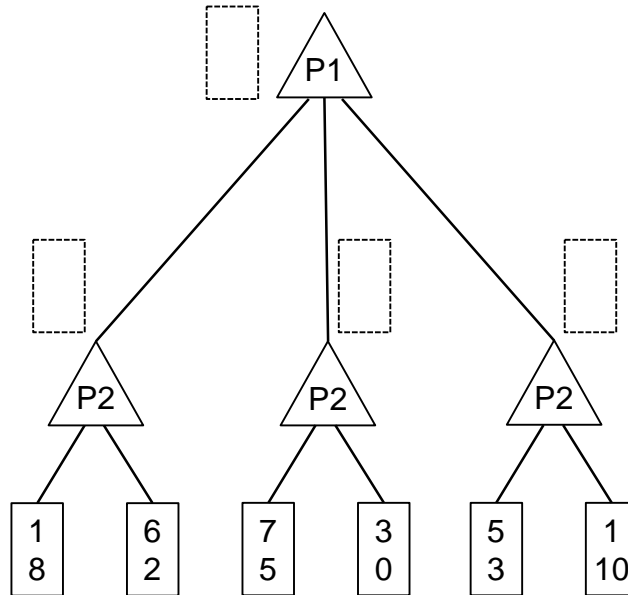


Q1. Games



For the following game tree, each player maximizes their respective utility. Let x, y respectively denote the top and bottom values in a node. Player 1 uses the utility function $U_1(x, y) = x$.



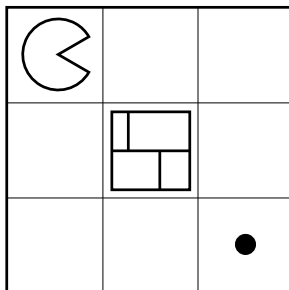
Both players know that Player 2 uses the utility function $U_2(x, y) = x - y$.

- (a) Fill in the rectangles in the figure above with pair of values returned by each max node. From top-down, left-right: $(6, 2), (6, 2), (3, 0), (5, 3)$
- (b) You want to save computation time by using pruning in your game tree search. On the game tree above, put an 'X' on branches that do not need to be explored or simply write 'None'. Assume that branches are explored from left to right. None.

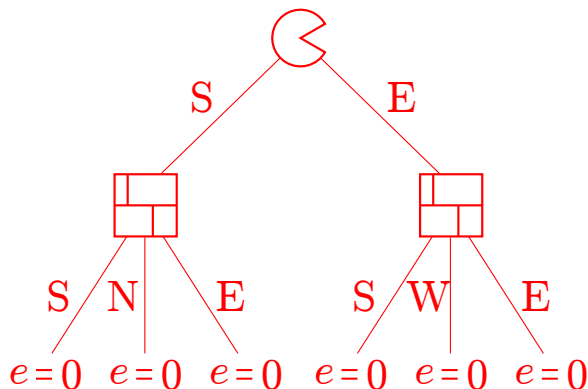
Q2. Surrealist Pacman

In the game of Surrealist Pacman, Pacman  plays against a moving wall . On Pacman's turn, Pacman must move in one of the four cardinal directions, and must move into an unoccupied square. On the wall's turn, the wall must move in one of the four cardinal directions, and must move into an unoccupied square. The wall cannot move into a dot-containing square. Staying still is not allowed by either player. Pacman's score is always equal to the number of dots he has eaten.

The first game begins in the configuration shown below. Pacman moves first.



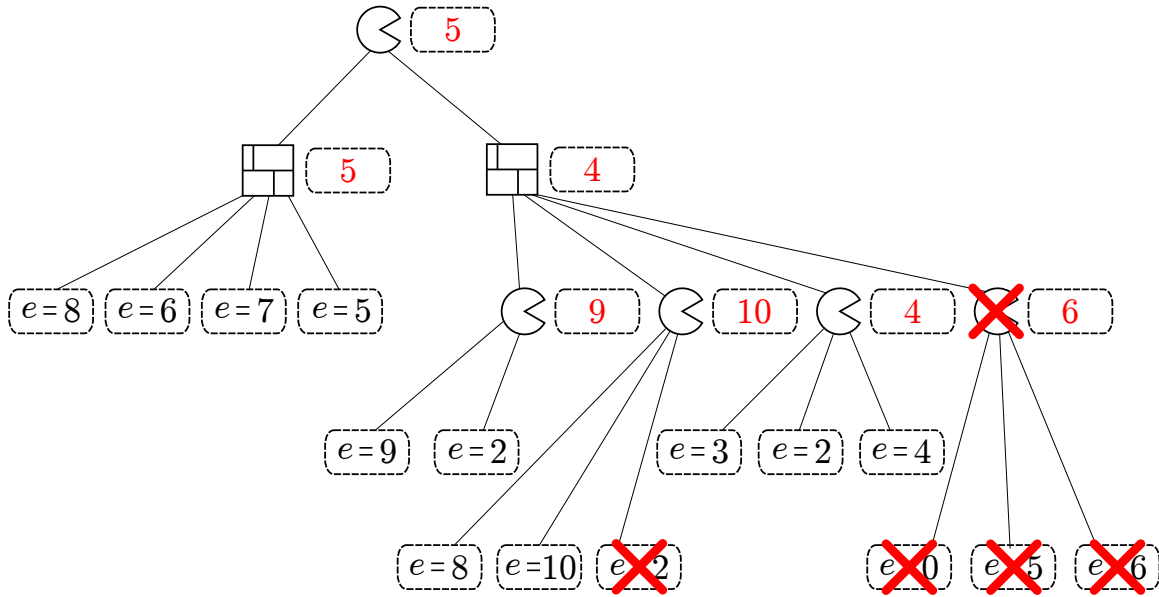
- (a) Draw a game tree with one move for each player. Nodes in the tree represent game states (location of all agents and walls). Edges in the tree connect successor states to their parent states. Draw only the legal moves.



- (b) According to the depth-limited game tree you drew above what is the value of the game? Use Pacman's score as your evaluation function.
0. All leaves have value 0.
- (c) If we were to consider a game tree with ten moves for each player (rather than just one), what would be the value of the game as computed by minimax?
1. Pacman can force a win in ten moves.

A second game is played on a more complicated board. A partial game tree is drawn, and leaf nodes have been scored using an (unknown) evaluation function e .

- (d) In the dashed boxes, fill in the values of all internal nodes using the minimax algorithm.
- (e) Cross off any nodes that are not evaluated when using alpha-beta pruning (assuming the standard left-to-right traversal of the tree).



Running alpha-beta pruning on the game tree.

Root: $\alpha = -\infty, \beta = \infty$

-- Left wall: $\alpha = -\infty, \beta = \infty$

---- Leaf node: $e = 8$. Propagate $e = 8$ back to parent.

-- Left wall: Current value is 8. $\alpha = -\infty, \beta = 8$. Max doesn't have a best value, so continue exploring.

---- Leaf node: $e = 6$. Propagate $e = 6$ back to parent.

-- Left wall: Current value is 6. $\alpha = -\infty, \beta = 6$. Max doesn't have a best value, so continue exploring.

---- Leaf node: $e = 7$. Propagate $e = 7$ back to parent.

-- Left wall: No update. Current value is 6. $\alpha = -\infty, \beta = 6$. Max doesn't have a best value, so continue exploring.

---- Leaf node: $e = 5$. Propagate $e = 5$ back to parent.

-- Left wall: Current value is 5. We're done here, so propagate 5 to root.

Root: Current value is 5. $\alpha = 5, \beta = \infty$. Explore right.

-- Right wall: $\alpha = 5, \beta = \infty$.

---- 1st Pac: $\alpha = 5, \beta = \infty$

----- Leaf node: $e = 9$. Propagate $e = 9$ back to parent.

---- 1st Pac: Current value is 9. $\alpha = 9, \beta = \infty$ MIN doesn't have a best value, so continue exploring.

----- Leaf node: $e = 2$. Propagate $e = 2$ back to parent.

---- 1st Pac: No change. Current value is 9. Propagate 9 to parent.

-- Right wall: Current value is now 9. $\alpha = 5, \beta = 9$. MIN wants anything less than 9 at this point, but it's still possible for MAX to get more than 5. Continue exploring.

---- 2nd Pac: $\alpha = 5, \beta = 9$

----- Leaf node: $e = 8$. Propagate $e = 8$ back to parent.

---- 2nd Pac: Current value is now 8. $\alpha = 8, \beta = 9$. Again, still possible for both players to benefit (Imagine value = 8.5). Continue exploring.

----- Leaf node: $e = 10$. Propagate $e = 10$ back to parent.

---- 2nd Pac: Current value is now 10. So now, we know that $v > \beta$, which means that one of the players is going to be unhappy. MAX wants something more than 10, but MIN is only satisfied with something less than 9, so we don't have to keep exploring.

---- *PRUNE* $e = 2$.

---- 2nd Pac: returns value of 10 to parent.

-- Left Wall: No change in value, current value is still 9. $\alpha = 5, \beta = 9$. Again, still possible for both players to benefit, so continue exploring.

---- 3rd Pac: $\alpha = 5, \beta = 9$

----- Leaf node: $e = 3$. Propagate $e = 3$ back to parent.

---- 3rd Pac: Current value is 3. $\alpha = 5, \beta = 9$. Continue exploring.

----- Leaf node: $e = 2$. Propagate $e = 2$ back to parent.

---- 3rd Pac: No change in value. Current value is 3. $\alpha = 5, \beta = 9$. Continue exploring.

----- Leaf node: $e = 4$. Propagate $e = 4$ back to parent.

---- 3rd Pac: Current value is 4. We're done, so return value of 4 to parent.

-- Left Wall: Current value becomes 4. At this point, we know that MIN wants anything that is less than or equal to 4. However, MAX is only satisfied with something that is 5 or greater. Hence, we don't need to explore the rest of the children of this node since MAX will never let the game get down to this branch.--

Prune rest

(Filling values returned by alpha-beta or not crossing off children of a crossed off node were not penalized.)

Suppose that this evaluation function has a special property: it is known to give the correct minimax value of any internal node to within 2, and the correct minimax values of the leaf nodes exactly. That is, if v is the true minimax value of a particular node, and e is the value of the evaluation function applied to that node, $e - 2 \leq v \leq e + 2$, and $v = e$ if the node is a dashed box in the tree below.

Using this special property, you can modify the alpha-beta pruning algorithm to prune more nodes.

- (f) Standard alpha-beta pseudocode is given below (only the max-value recursion). Fill in the boxes on the right to replace the corresponding boxes on the left so that the pseudocode prunes as many nodes as possible, taking account of this special property of the evaluation function.

```

function MAX-VALUE(node,  $\alpha$ ,  $\beta$ )
   $e \leftarrow$  EVALUATIONFUNCTION(node)
  if node is leaf then
    return  $e$ 
  end if
   (1)
   $v \leftarrow -\infty$ 
  for child  $\leftarrow$  CHILDREN(node) do
     (2)
    if  $v \geq \beta$  then
      return  $v$ 
    end if
     $\alpha \leftarrow$  MAX( $\alpha$ ,  $v$ )
  end for
  return  $v$ 
end function
  
```

Fill in these boxes:

(1)

```

if  $e - 2 \geq \beta$  then
  return  $e - 2$ 
end if
  
```

(2)

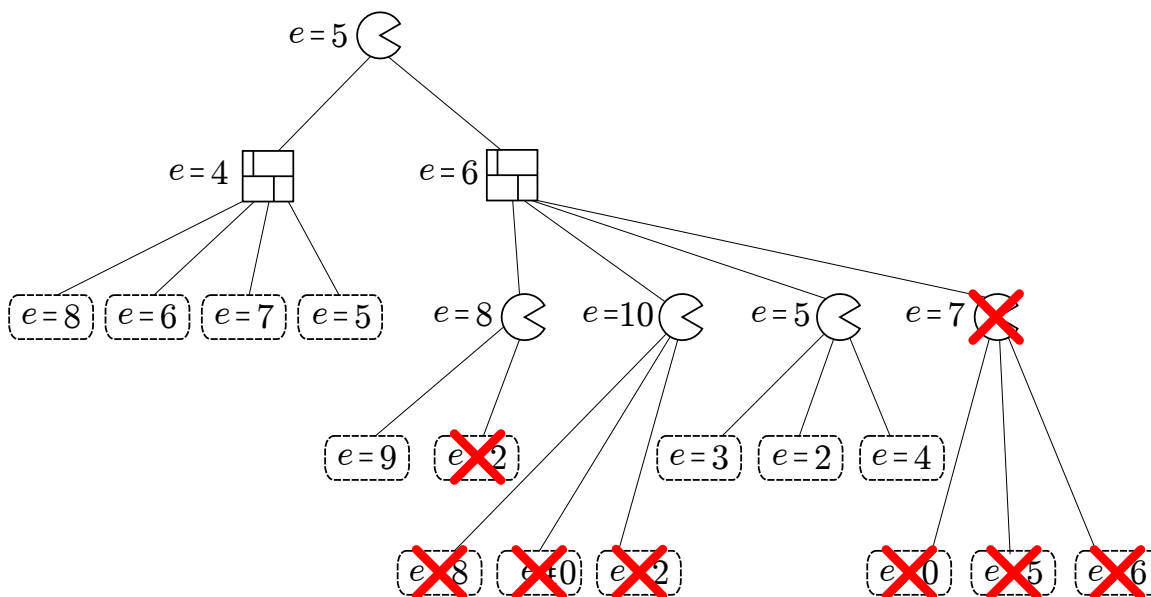
```

 $v \leftarrow$  MAX( $v$ , MIN-VALUE(child,
  MAX( $\alpha$ ,  $e - 2$ ), MIN( $\beta$ ,  $e + 2$ )))
  
```

(Variations are possible.)

The same game tree is shown below, with the evaluation function applied to *internal* as well as leaf nodes.

- (g) In the game tree below cross off any nodes that can be pruned assuming the special property holds true. If not sure you correctly formalized into pseudo-code your intuition on how to exploit the special property for improved pruning, make sure to annotate your pruned nodes with a brief explanation of why each of them was pruned.



Running pruning on game tree in detail. W1 and W2 refer to the left and right wall nodes respectively. P1, P2, P3, P4 refer to Pacman nodes on the third level, left to right.

Root: Evaluation function returns 5. Range of value: [3, 7]. Set $\alpha : 3, \beta : 7$ and explore(W1, α, β).

frm[o]-W1 Evaluation function returns 4. Range of value: [2,6]. Set $\alpha : 3, \beta : 6$.

frm-eLeaf node $e = 8$. Send 8 back to W1.

frm[o]-W1 $v = 8$. $v < \alpha$? No. This means that MAX might still prefer this path.

frm-eLeaf node $e = 6$. Send 6 back to W1.

frm[o]-W1 $6 < 8$ so $v = 6$. $v < \alpha$? No. Continue exploring.

frm-eLeaf node $e = 7$. Send 7 back to W1.

frm[o]-W1 $7 > 6$, so no change, $v = 6$. $v < \alpha$? No. Continue exploring.

frm-eLeaf node $e = 5$. Send 5 back to W1.

frm[o]-W1 $5 < 6$, so $v = 5$. Done. Send 5 back to root.

Root: Gets value 5, so $v = 5$. $\alpha : \max(3, 5) = 5, \beta : 7$. Still exploring right branch since MAX could get a value $5 \leq v \leq 7$. Explore(W2, α, β).

frm[o]-W2 Evaluation function returns 6. Range of values [4, 8]. $\alpha : 5, \beta : 7$, explore(P1, α, β).

frm-eP1 Evaluation function returns 8. Range: [6, 10], $\alpha : 6, \beta : 7$.

- - - Leaf node: $e = 9$. Send $e = 9$ back to P1.

frm-eP1 $v = 9$. $v > \beta$? *Yes!*. We can prune here since P1 wants any value > 9 . However, at the root we know that the maximum value that MAX can get is 7. Hence, there is no way the game can get down to P1. (Meaning that the value at the root can not be 9).

- - - Leaf node: Prune $e = 2$. Return 9 to W2.

frm[o]-W2 $v = 9$. α, β don't change: $\alpha : 5, \beta : 7$. Explore(P2, α, β).

frm-eP2 Evaluation function returns 10. Range [8, 12], $\alpha : 8, \beta : 7$. Notice that the best value for MIN that can be achieved at P2 is 8. However, the best value for MIN at the root is 7. Hence, there's no way the game can get down to P2. (Meaning the value of the root can not be 8). *Prune all of P2's children!*. We can return 8 to W2 since we know that there is some other path through W2 that yields a reward ≤ 7 .

frm[o]-W2 $v : \min(8, 9) = 8, \alpha : 5, \beta : 7$. $v < \alpha$? No! Explore(P3, α, β).

frm-eP3 Evaluation function returns 5. Range: [3, 7], $\alpha : 5, \beta : 7$.

- - - Leaf node: $e = 5$. Send 3 back to P3.

frm-eP3 $v = 3$. $v > \beta$? No! Meaning MIN might still prefer this branch. $\alpha : 5, \beta : 7$.

- - - Leaf node: $e = 2$. Send 2 back to P3.

frm-eP3 $v = 3$. $v > \beta$? No! $\alpha : 5, \beta : 7$.

- - - Leaf node: $e = 4$. Send 4 back to P3.

frm-eP3 $v = 4$. Done. Return 4 to W2.

frm[o]-W2 $v : \min(8, 4) = 4, \alpha : 5, \beta : 7$. $v < \alpha$? *Yes!* Since MAX can guarantee a value of 5 and MIN will only accept something < 4 , don't need to explore any further. *Prune P4 and all its children*. Return 4 to root.

Root: *Done*.